**FORSCHUNGSZENTRUM JÜLICH GmbH**
**Zentralinstitut für Angewandte Mathematik**
**D-52425 Jülich, Tel. (02461) 61-6402**

Interner Bericht

# Partitioning of Hypercubes by Resolution of Combinatorial Designs

***Heribert C. Burg***

KFA-ZAM-IB-9619

Dezember 1996
(Stand 23.12.96)

Dieser Bericht wurde zur Publikation eingereicht.

# Partitioning of Hypercubes by Resolution of Combinatorial Designs

**Heribert C. Burg**

**Central Institute for Applied Mathematics,
Research Centre Jülich, 52425 Jülich, Germany**

**E-Mail: h.burg@kfa-juelich.de**

**Abstract:**

A method for the partitioning of multiprocessors with a static interconnection network in hypercube topology is presented. It is based on the view of a static interconnection network as a combinatorial design. Our mere interest is to show that by means of a transformation, well-established methods of combinatorial design theory can be used to manage tasks emerging from parallel processing, rather than to give a highly efficient hardware-related partitioning technique.

**Keywords:**

Interconnection networks, parallel processing, combinatorial designs, partitioning, hypercube.

## 1 Introduction

Many articles in the literature describe the use of (simple) graphs for a wide variety of applications in computer science. In this context, hypergraphs and especially combinatorial designs are considered only in a few papers [3]. Concentrating on the field of parallel processing a lot of work still has to be done. There has been research mainly in the construction of multiprocessor networks using balanced incomplete block-designs, cf. section 2, or pairwise balanced designs [1]. The spectrum of possible applications for combinatorial design methods in parallel processing, however, is much broader [2].

Considering parallel architectures and parallel algorithms as combinatorial designs, methods from combinatorial mathematics can be used as new, genuine solution concepts for tasks emerging in parallel processing like mapping, partitioning, embedding, and routing. We will focus here on the task of partitioning a (massively) parallel computer.

To exploit system resources efficiently, computers are normally used in multiprogramming mode: At the same time several programs are executed simultaneously in the system. In a monoprocessor system, for instance, multiprogramming allows the use of the computing power of the processor by one program, while another program is doing I/O. Multiprogramming becomes even more important for multiprocessor systems, as it is getting more complicated to exploit a satisfactory fraction of the combined computing power. For the multiprogramming mode in a multiprocessor system, the entire system has to be divided into several *partitions* to provide parallel users with their own work space.

Since for massively-parallel systems the principle of shared real memory is technically not feasible — we do not consider here systems with shared virtual memory —, communication is done via message passing. In most multiprocessor systems, the costs of communication increase with the distance of the communicating processors. Therefore, a partition to be created should consist of processors being very close together forming a connected *cluster*.

Some algorithms match with special topologies of multiprocessor systems (with static interconnection networks). For example, the communication structure of the radix-2-FFT is optimally reflected by the connection structure of the hypercube. Therefore, the partition should represent the same topology as the whole system, in case the user is aiming at the whole system hoping to get it exclusively. On the other hand, it can generally be observed that a certain topology is best divided into substructures of the same kind. In the context of partitioning, optimality is considered in terms of a maximum number of usable processors. Hence, in the following **partitioning** means the division of a multiprocessor system of a certain topology into several partitions of the same topology.

In the next section, we briefly introduce the combinatorics of experimental designs and their relation to interconnnection networks. In the third section, it is shown how hypercubes can be partitioned by the use of designs, and the last section contains several concluding remarks.

## 2 Combinatorial Designs

We start by giving some definitions necessary to introduce combinatorial designs; further mathematics on combinatorial designs can be taken from text books on this topic, e.g.

[6]. In the context addressed here, the definition of a combinatorial design as a pair of parameters is sufficient; sometimes it can be required or more elegant to define a design as a triple.

Let $V$ be a finite set of cardinality $v$. A **set system $\mathcal{B}$ on** $V$ is a collection of subsets of $V$. Elements of $\mathcal{B}$ are called **blocks**; the number of blocks is denoted by $b =\mid \mathcal{B} \mid$. Let $V$ be a finite set and $\mathcal{B}$ a set system on $V$. Then the pair $(V, \mathcal{B})$ is called a **(combinatorial) design**.

The **replication number** $r_x$ of an element $x \in V$ is defined by the number of blocks containing $x$. The design is called **symmetric** if $v = b$. If all blocks of the combinatorial design have the same number of elements, say $k$, the design is called $k$**-uniform**. The design is called **complete** if each block contains all elements of $V$.

There are many interesting properties around designs; we will here just point out the property of *balance*. Choosing any subset $S \subseteq V$, one can look whether $S$ is a subset of any of the blocks of $\mathcal{B}$. The number of blocks of $\mathcal{B}$ each complete containing $S$ is called **the index of** $S$ **within** $\mathcal{B}$. It is written as $\lambda(S) \geq 0$. Looking at all subsets of $V$ with the same number of elements, say $t$, the (possibly different) numbers of blocks containing these subsets form a set $\Lambda_t$ called $t$**-index set of** $\mathcal{B}$; $\Lambda_t := \{\lambda(S) :\mid S \mid= t, S \subseteq V\}$.

A set-sytem is called $t$**-balanced**, if its $t$-index set contains exactly one element $\lambda_t$, satisfying $\lambda_t > 0$. A 1-balanced set system just means that each element of $V$ occurs equally often within the blocks of $\mathcal{B}$. Every set system is 0-balanced, with $\lambda_0$ being the number of blocks. A $t$-balanced $k$-uniform set system is $(t-1)$-balanced as well. Look at the following design example $(V, \mathcal{B})$: $V = \{0, 1, 2, 3, 4, 5, 6\}$, $B = \{(0, 1, 2), (2, 3, 4), (4, 5, 0), (0, 6, 3), (1, 6, 4), (2, 6, 5), (1, 3, 5)\}$. This combinatorial design is 2-balanced, since each 2-element-subset of $V$ appears equally often (once) within blocks of the set system $\mathcal{B}$.

With these definitions we come to the most important structure within combinatorics of designs: A **balanced incomplete block-design (BIBD)** is a pair $(V, \mathcal{B})$, with $\mathcal{B}$ being a $k$-uniform set system on $V$, which is 2-balanced with $\lambda := \lambda_2$. A BIBD $(V, \mathcal{B})$ will be described as $(v, k, \lambda)$-design.

For a design $(V, \mathcal{B})$, a **parallel class** of blocks $\mathcal{P} \subseteq \mathcal{B}$ is a set of blocks such that no two blocks intersect and the union of all blocks of $\mathcal{P}$ is $V$. If $\mathcal{B}$ can be split into parallel classes, this decomposition is called **resolution**, and $(V, \mathcal{B})$ is called a **resolvable design**.

We restrict our work here to static interconnection networks. With respect to this, an interconnection network consists of processors (nodes) and connections between them. A connection will be called **link**, if it interconnects exactly two nodes; a connection joining

three or more nodes is called **bus**. An interconnection network of a multiprocessor can be regarded as a combinatorial design, if the compute nodes are taken as the elements of the set $V$, while the interconnections between them are described as blocks. For example, a bus which connects the nodes 1, 4 and 7, is then denoted as block $(1, 4, 7) \in \mathcal{B}$.

Simple graphs are not appropriate to model interconnection networks containing busses, as graphs are 2-uniform restrictions of combinatorial designs, and may therefore just serve in modelling systems with links. Static parallel algorithms can be described by designs in an analoguous way, if they are considered as sets of separately executable modules. If different phases with different communication geometries exist, each phase is represented by its own design. Based on these relations, many statements and methods can be inferred from the mathematics of combinatorial designs to the tasks within parallel processing.

## 3 Partitioning of Hypercubes

The hypercube is one of the few topologies which many hardware implementations are based on; thus it is certainly one of the most interesting topologies to deal with. Additionally, some effects of partitioning and of other operations on parallel structures can be observed better by using a more complex structure like the hypercube than using rings, stars or grids. Consequently, parallel computer related research has often been focussed on the hypercube architecture [4,5].

The general problem, whether an accidental request for an accidental number of subcubes of various sizes can be met, is NP-complete. Therefore, partitioning of hypercubes is done heuristically. Known strategies are linear strategy, Buddy-strategy, and Gray-Code strategies. Besides efficiency, a good strategy should guarantee that if there is a subcube of sufficient size it is indeed given to the user. The problem of *fragmentation* proves difficult to manage. Among the remedies to cope with this problem, there is the expensive technique of migrating a job to another subcube.

We will not concentrate on these technical problems here; our concern is to present a partitioning strategy which is derived from combinatorics, based on the fact that a multiprocessor computer with static interconnection topology — as well as a parallel algorithm — can be regarded as a combinatorial design.

***Example:***

A hypercube of dimension 4, cf. figure 1, corresponds to the following design $(V, \mathcal{B})$:

$V = \{0, 1, 2, ..., 15\}$, $\mathcal{B} = \{(0, 15), (1, 14), (2, 13), ..., (10, 11), (12, 13), (14, 15)\}$

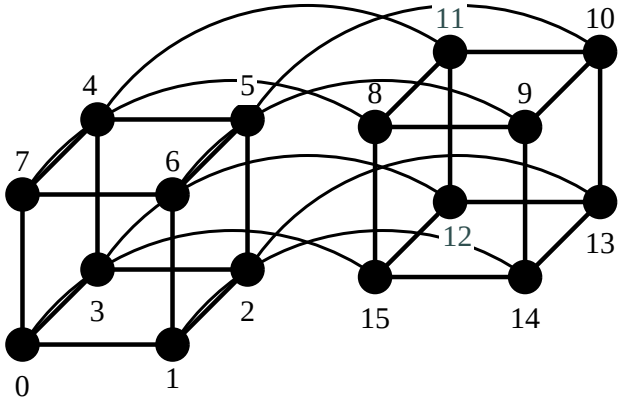


Figure 1. 4-dimensional hypercube as a combination of two 3-dimensional hypercubes

Using each link in both directions, the link (a,b) is equal to the link (b,a). Taking the "<"-order of numbers as the basis, the lexicographic order " $\prec$ " on pairs of numbers induces the following order into the set system $\mathcal{B}$:

$(0,1) \prec (0,3) \prec (0,7) \prec (0,15) \prec (1,2) \prec (1,6) \prec (1,14) \prec (2,3) \prec (2,5) \prec (2,13) \prec (3,4) \prec (3,12) \prec (4,5) \prec (4,7) \prec (4,11) \prec (5,6) \prec (5,10) \prec (6,7) \prec (6,9) \prec (7,8) \prec (8,9) \prec (8,11) \prec (8,15) \prec (9,10) \prec (9,14) \prec (10,11) \prec (10,13) \prec (11,12) \prec (12,13) \prec (12,15) \prec (13,14) \prec (14,15)$

The design $(V, \mathcal{B})$ will prove to be resolvable. If, in an attempt to resolve the design, the blocks of the set system $\mathcal{B}$ are attached to parallel classes $PC^i$ (by opening new classes only if needed and) by preferring the class with the smallest possible superscript, the unique result of the resolution will be the following classes:

$PC^1 = \{(0,1),(2,3),(4,5),(6,7),(8,9),(10,11),(12,13),(14,15)\}$

$PC^2 = \{(0,3),(1,2),(4,7),(5,6),(8,11),(9,10),(12,15),(13,14)\}$

$PC^3 = \{(0,7),(1,6),(2,5),(3,4),(8,15),(9,14),(10,13),(11,12)\}$

$PC^4 = \{(0,15),(1,14),(2,13),(3,12),(4,11),(5,10),(6,9),(7,8)\}$

| | Gray-Code table | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| address | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| code | 0000 | 0001 | 0011 | 0010 | 0110 | 0111 | 0101 | 0100 |
| address | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| code | 1100 | 1101 | 1111 | 1110 | 1010 | 1011 | 1001 | 1000 |

Table 1. Gray code table for the numbers 0,...,15

Following the nodes in the basic (entire) hypercube in figure 1 in the order of the Gray code, cf. table 1, the first subcube (of dimension 3) to be produced consists of the nodes

{0,1,2,3,4,5,6,7}. As a consequence, the second subcube to split contains the nodes {8,9,10,11,12,13,14,15}. Looking for the interconnections of the basic cube by which the two cubes of dimension 3 are fitted together to form a hypercube of dimension 4, these are the connections {(0,15), (1,14), (2,13), (3,12), (4,11), (5,10), (6,9), (7,8)}. These somewhat artificially introduced interconnections are however exactly the same interconnections as those collected by the parallel class $PC^4$. If the (lexicographically generated) order of the set system $\mathcal{B}$ is transferred to the parallel classes created, one can construct the two subcubes by the parallel classes $PC^1, PC^2$ and $PC^3$. The first subcube is built using the first four blocks out of every class with respect to the order, and the second subcube is built using the last four blocks. Thus the process of partitioning is complete. This correspondance is one of many interesting relations between combinatorics of designs and parallel processing.

The extension to the general case results in the following theorem and corollary.

***Theorem:***

A design $(V, \mathcal{B})$ reflecting a hypercube multiprocessor, which is constructed according to the Gray code, is always resolvable. The number of parallel classes is equal to the dimension of the hypercube.

***Proof:***

Let us consider a hypercube of dimension $n$ with $N = 2^n$ nodes. W.l.o.g. the hypercube nodes have been numbered from 0 to $N - 1$ and ordered within the blocks by the linear ordering "<". The ordering of the nodes within the blocks induces a lexicographical ordering " $\prec$ " of the blocks within the set system $\mathcal{B}$. Let the hypercube be addressed according to the Gray code as provided.

The proof is done by induction on the dimension of the hypercube, and on the construction principle of the Gray code, respectively.

I. Induction basis (for $n = 2$)

*To show*: A hypercube of dimension 2 is resolvable. The resolution creates two parallel classes.

A hypercube of dimension 2 is represented by the following design $(V, \mathcal{B})$ with $V = \{0, 1, 2, 3\}$ and $\mathcal{B} = \{(0, 1), (0, 3), (1, 2), (2, 3)\}$. The partitioning of the hypercube into the classes $C^1 = \{(0, 1), (2, 3)\}$ and $C^2 = \{(0, 3), (1, 2)\}$ is a resolution of the hypercube, because $C^1$ and $C^2$ are parallel classes as they are sets of disjoint blocks, which are containing all elements of V, and the union of the two classes $C^1$ and $C^2$ gives the whole set system $\mathcal{B}$.$\diamondsuit$

II. Induction step (conclusion from $n$ to $n+1$)

*Hypothesis*: A hypercube of dimension $n$ is resolvable.

*Statement*: A hypercube of dimension $n+1$ is resolvable.

*Proof*:

Given a hypercube of dimension $n$. Let by hypothesis $PC^1, ..., PC^n$ be a resolution of the hypercube of dimension $n$. Now the hypercube is duplicated. Then nodes with the same addresses (numbers) are connected. Thus a new hypercube of dimension $n+1$ is created. Now all node addresses of the entire structure are extended by 1 bit, placing a 0 in front of the addresses of the one subcube and a 1 in front of the addresses of the other. So each address in the entire construction is unique, as the addresses of each of the two components were unique. The new extended addressing is again according to the Gray code.

The $N$ addresses $(0a_{i,n-1}...a_{i,0}, 1a_{i,n-1}...a_{i,0})$ for $i = 0, ..., n-1$ define those $N = 2^n$ connections, which combine both cubes of dimension $n$ to the new cube of dimension $n+1$. Using the induction hypothesis we begin with the resolution $\zeta$ of the hypercube of dimension $n$: $\zeta = \{PC^1, ..., PC^n\}$. Now each parallel class $PC^i$ containing $N$ blocks is extended to $2N$ blocks replacing each block of the form $(a_{j,n-1}...a_{j,0}, a_{k,n-1}...a_{k,0})$ by 2 blocks of the form $(0a_{j,n-1}...a_{j,0}, 0a_{k,n-1}...a_{k,0})$ and $(1a_{j,n-1}...a_{j,0}, 1a_{k,n-1}...a_{k,0})$. This results in the following situation:

- Let us name those classes of $2N$ blocks, which have been generated by the replacement of the parallel classes of $\zeta$ as $C^1, ..., C^n$ and name the introduced interconnections between the two components as (class) $C^{n+1}$. Then we find that these $n+1$ classes form a *decomposition* of the hypercube of dimension $n+1$, because they contain all interconnections belonging to it.
- Each of the created classes $C^i, i = 1, ..., n+1$, is a *parallel* class, because

  (a.) the replacement operation on the parallel classes of the $n$-dimensional cube keeps the *uniqueness* of the elements in each class; instead of the address $a_{i,n-1}...a_{i,0}$ now the two addresses $0a_{i,n-1}...a_{i,0}$ and $1a_{i,n-1}...a_{i,0}$ are in the new class. Also class $C^{n+1}$ contains every address exactly once, which is guaranteed by the construction principle of the interconnections between the components. In other words: Classes $C^1, ..., C^{n+1}$ are sets of *disjoint* blocks.

  (b.) (additionally) each class $C^i$ contains *all* nodes of the $(n+1)$-dimensional hypercube. Following the induction hypothesis, we started with a resolution of the $n$-dimensional hypercube, which guarantees by definition that each of its classes contains all elements of the corresponding hypercube. By the replacement

operation according to the construction principle of the Gray code, it is achieved that each of the new classes $C^1, ..., C^n$ contains each node of the duplicated hypercube. The construction method for the combined cube also guarantees that each of its nodes can be found in the class of the introduced interconnections $C^{n+1}$.

Hence, the classes $C^1, ..., C^{n+1}$ form a resolution of the hypercube of dimension $n+1$. q.e.d.

***Corollary:***

Following the construction principle used in the proof of the resolvability, it can be shown by induction that the two partitions to be constructed can be taken from the first (second, respectively) half of the blocks of the classes $C^1, ..., C^n$ while the blocks of the class $C^{n+1}$ are to be neglected.

## 4 Concluding Remarks

A lower bound for the number of parallel classes needed for the resolution of a hypercube of dimension $n$ is the replication number of the corresponding combinatorial design $r_x = n$. Since a parallel class consists of disjoint blocks and a resolution delivers disjoint parallel classes, $r_x = n$ enforces at least $n$ parallel classes. Carrying out a resolution instead of an ordinary decomposition makes $n$ also an upper bound. The lexicographical ordering is necessary, because it guarantees the uniqueness of the parallel classes. If on a massively parallel computer in multiprogramming mode several subcubes of different dimensions are needed, one or both of the subcubes generated can be further partitioned successively. Other combinatorial design methods serve for the partitioning of other topologies, e.g. trees or meshes.

## 5 Literature


[1] S. Y. Berkovich, Multiprocessor Interconnection Network Using Pairwise Balanced Designs, *Information Processing Letters* **50** (1994), pp. 217–222

[2] H. C. Burg, Anwendungen kombinatorischer Versuchspläne in der Parallelverarbeitung (in German), Dissertation, Technical University of Aachen, Report Jül-2913, Research Centre Jülich, Jülich 1993

[3] C. J. Colbourn and P. C. van Oorschot, Applications of Combinatorial Designs in Computer Science, *ACM Computing Surveys* **21** (1989), pp. 223–250

[4] D. Das Sharma and D. K. Pradhan, Processor Allocation in Hypercube Multicomputers: Fast and Efficient Strategies for Cubic and Noncubic Allocation, *IEEE Transactions on Parallel and Distributed Systems* **6** (1995), pp. 1108–1123

[5] S. Rai, J. L. Trahan, and T. Smailus, Processor Allocation in Hypercube Multiprocessors, *IEEE Transactions on Parallel and Distributed Systems* **6** (1995), pp. 606–616

[6] W. D. Wallis, *Combinatorial Designs*, Marcel Dekker, New York 1988